IHFS Quality Code Operations Guide

National Weather Service
Office of Hydrologic Development
September 29, 2000

Table of Contents

1. Introduction

2. Description of QC Operations

3. Structure of Quality Code

4. Programmers Reference Information

Appendix A. Diagram of QC Operations in the IHFS

Appendix B. Example Displays from QC Related Applications

1. Introduction

Beginning with Release 5.0 of AWIPS, the Integrated Hydrologic Forecast System (IHFS) database structure and the WFO Hydrologic Forecast System (WHFS) applications have incorporated a comprehensive set of data quality operations. Associated with each observed and forecast value in the database is a quality control code which indicates the quality of the value based on external and internal tests of the value. Depending on the value of the quality code, the data are handled in different ways within the IHFS data flow and applications may or may not use the data.

1.1 Three Level QC Model

In the design of the QC infrastructure, a key issue was how to summarize the quality of a data value in very succinct terms. The adopted method is to use a three-level system, where every value is designated and treated as being either GOOD, QUESTIONABLE, or BAD. A value is assumed to be GOOD unless otherwise determined; i.e. GOOD is the default quality level. Based upon external information or internal tests, a value may be designated as BAD, which implies that the value should not be used under any circumstances. A value may be designated as QUESTIONABLE, which implies that something “suspicious” was noted with the value, but not with such certainty that it can be assumed to be unusable. QUESTIONABLE values should probably be reviewed further by manual means which can then designate the value as being GOOD, if appropriate.

Note that although a value is summarized as one of these three levels, the full details of all the quality tests can be obtained via inspection of specific components of the quality control code. As will be described later, the three-level characterization of each value allows applications to quickly and easily determine whether or not to use a value.

1.2 Local Applications

Another objective of the design of the QC processing is to ensure that the structure of the IHFS database facilitates the incorporation of local applications. The IHFS database and WHFS applications are designed to be adaptable to external data and user interfaces, with room to expand. While the WHFS applications provide significant support in managing the quality code, they will never be able to satisfy all the varied needs of the NWS offices. Local offices are encouraged to develop or modify local applications to assess the data quality and set the quality code accordingly, and to read the quality code and use the data in the appropriate manner. Local applications should follow the framework governing the QC operations discussed in Section 2, and should adhere to the structure of the quality code field described in Section 3. Software is available to support QC operations; this software is documented in Section 4.

1.3 Overlap of QC and Alert/Alarm Processing

In the WHFS there is processing related to quality control and there is processing for monitoring data that may exceed some alert or alarm thresholds. Operations for both of these features require an application to compare the data value(s) with some predefined threshold. When processing data, it makes sense from a speed/performance standpoint to perform the QC and alert/alarm operations together. Both require the physical data and the threshold limits data to be read and compared. In practice, this takes an amount of time which is time best spent if done simultaneously. Even though the applications mentioned later are also performing alert/alarm checking, this document only describes the quality control operations. The alert/alarm operations are described in a separate document.

1.4 Document Overview

Section 2 of this document describes the data processing in terms of the quality code. It covers the overall system approach for handling the quality code operations within the WHFS. Section 3 provides a detailed description of the internal structure of the quality code field in the database. The document concludes with Section 4, which contains programmers information on how to manage the quality code with regard to reading, setting, or modifying its value.

Appendix A includes an invaluable diagram which shows the path of data through the IHFS database and WHFS applications, again in terms of the quality code. This diagram is heavily referenced in Section 2. Appendix B gives example screen displays from the primary user interfaces in the WHFS applications which relate to the quality code.

2. Description of QC Operations

Appendix A contains a diagram which depicts the overall processing within the WHFS that pertains to the quality control operations. This section uses this diagram as the focus for describing the various tests, data paths, and user interactions that can occur. Each component of the diagram is described in detail within this section, in addition to discussing some other related topics.

2.1 SHEF Processing

For the most part, data “enters” the IHFS database via the SHEF decoder; internally generated data is discussed later. As shown at the top of the diagram, data are contained within SHEF products and are processed by the SHEF decoder application.

The SHEF data are first decoded. Each decoded value has a set of SHEF attributes associated with it, including the SHEF physical element, duration, type-source, extremum and qualifier code. The SHEF decoder also associates a new field with the value, after the value is decoded. The new field is the IHFS quality_code, which has an initial, i.e. default, value assigned to it, indicating a quality of GOOD. This quality_code field is separate from the SHEF qualifier code, although they are related as is discussed below.

2.2 SHEF Qualifier Code

One of the decoded attributes associated with every value is the SHEF qualifier code. The current SHEF handbook - SHEF Version 1.3, dated March 1998 - lists the possible values for the SHEF qualifier code in Table 10. Since the document was published, a few additions have been made to the set of possible values. For the sake of completeness, the entire set of code values, including the newly adopted values, is listed in Figure 1. Note that the “levels” referred to in the SHEF qualifier code descriptions are used somewhat arbitrarily by the external providers of SHEF products and do not conform to any formal convention.

After setting the initial default value of the IHFS quality code, the external SHEF qualifier code is inspected. This code has 12 possible values: five indicate the value is GOOD; two indicate the value is QUESTIONABLE; two indicate the value is BAD; the remaining three indicate that the value was not tested or the SHEF qualifier code has nothing to do with quality control. If the SHEF qualifier code indicates the data are BAD or QUESTIONABLE, then the IHFS quality code is updated to note this.

Value	Description		
(GOOD)		(BAD)	
G	Good, Manual QC *	B	Bad, Manual QC *
M	Manual Edit *	R	Rejected by Level1
S	Screened Level1		
V	Verified Level1, Level2	(Unspecified/GOOD)	
P	Passed Level1, Level2, Level3 *	Z	No QC Performed
		E	Estimated
(QUESTIONABLE)		T	Triggered
F	Flagged By Sensor		
Q	Questioned in Level2, Level3	* = new	

Figure 1. SHEF Data Qualifier Codes

2.3 Single-Value Checks

Next, the single value checks are performed on the value. The SHEF decoder performs only single value checks because its entire operational concept is built around processing single values, which can be done relatively quickly. Checks which require multiple values, such as rate-of-change checks or spatial consistency checks, are more CPU-intensive.

The two single value checks currently provided are the gross range check and the reasonable range check. The gross range check determines if the value is within the specified maximum and minimum limits considered acceptable for the value. The reasonable range check operates in a very similar fashion, except that it uses a different set of maximum and minimum limits. The reasonable range is normally set to be within the range of the gross range, and is intended to reflect normal climatological extremes. For example, for air temperature data in New Hampshire, the gross range may be between -60 and 130 degrees, while the reasonable range may be between -40 and 115 degrees. If the value fails the gross range or reasonable range check, then the quality code is set to BAD or QUESTIONABLE, respectively.

2.4 Retrieval of Data Limits

The limits used for these checks, like all quality control limits, are stored in one of two tables, or possibly neither. Applications that require the limits first look in the LocDataLimits table for limits specified for the data value's location, physical element, duration, and time. The LocDataLimits table allows the user to define limits for a specific location. If a location-specific limit is found in this table, then it is used.

If a limit is not found, then a more general set of limits is searched. These limits are stored in the DataLimits table, which is read to see whether it has limits for the data value's physical element, duration, and time. The location is not considered when searching in this table; the DataLimits information is more general than the LocDataLimits information. If an entry for the data value's physical element, duration, and time is found in the DataLimits table, then the location-independent limit is used. The effect is that the LocDataLimits table allows the user to override the general location-independent data limits for given locations.

If neither a location-dependent (i.e. from LocDataLimits) nor a location-independent (i.e. from DataLimits) is found, then no limits are available and the limit tests are not performed. For the searches of both tables, note that the limits is given not just for the physical element, but also for the duration, and for the time of year. The time of year restriction allows seasonally-based limits to be defined.

2.5 Definition of Data Limits

The data limits in the tables LocDataLimits and DataLimits are managed by a user interface provided via the HydroBase interface. An example display of this table is given in Appendix B. The interface is not described here, but a brief discussion of special operational considerations is given.

In these two tables, a limit is assumed to be defined if a value is specified, as opposed to a "null", or blank, value being specified. Therefore, setting a limit to 0.0 does not deactivate the relevant test. To turn off a given test, a blank value must be specified which results in a null value being inserted into the database.

Note that when obtaining the limits for a given location, the applications will only use the limits in one of the two tables. If a user defines a given test's limit for a specific location, then a record will be created for that location in the LocDataLimits table, and that record may contain values for all limits, or for only some limits. Anytime an application needs a limit, it first looks in the location-specific record for that location. If the location is defined, but only some of its limits are specified, then for the other tests, it will not find the limit value, of course. In this case, the application will NOT attempt to search the general location-independent limits for the same physical element, duration, and time of year. This allows the user to avoid performing these other tests for a location, while still performing certain tests.

Also note that changes to the single-value checks will not take affect until the SHEF decoder application is restarted. This is because the SHEF decoder application buffers up the limit values in order to speed processing. However, changes to the rate-of-change thresholds will take effect the next time the rate-of-change checker application is executed.

2.6 Destination of Bad Data

The normal destination for data processed by the SHEF decoder is a set of tables referred to as the physical element tables. These are the primary tables in which all operational hydrometeorological data are stored. If a value is determined to be BAD, based on the interpretation of the external SHEF qualifier code or the internal single value tests, then the BAD value is written to one of two places, depending on a "switch" controlled by each office. A IHFS application token, shef_post_baddata, specifies whether the BAD data are written to the appropriate physical element table (shef_post_baddata = PE) or are written to the RejectedData "trashcan" table (shef_post_baddata = REJECT).

Each office must decide where to post this bad data. Writing the bad value to the physical element tables is desirable for those offices that wish to view the BAD data in the context of the possible surrounding GOOD data, for the same location and physical element. If data are BAD, the WHFS applications which process the data will not consider it; this includes RiverPro and the Precipitation Accumulation functions. BAD data are still considered by WHFS applications which simply display the data, such as the Time Series Data Viewer.

Other office may chose to send it to the RejectedData table, where it will still go unused by the applications. This keeps the data out of the physical element tables and therefore will not be available to any of the WHFS applications for processing or display purposes.

In both cases, the data are purged on a timed basis, although the retention criteria for the RejectedData table data is typically much smaller than that for the data in the physical element tables. Note there is a user interface to both sets of data, as discussed below.

2.7 Questionable/Bad Data Viewer

The WHFS HydroView application provides a user interface to list any QUESTIONABLE and BAD data that is co-mingled in the physical element tables. An example of this window is given in Appendix B. This interface does not list any GOOD data that is in the physical element tables; it only shows the QUESTIONABLE and BAD data. When selecting an item from the list, a descriptive phrase is displayed in the window that explains why the value is considered QUESTIONABLE or BAD; i.e. which check or test the value failed.

If the user wishes to see the full time series within which the QUESTIONABLE or BAD data is contained, the window provides the option to invoke the Time Series application. Doing this allows the data to be seen in its full context, which is helpful for identifying data trends, such as recurring data spikes.

Generally speaking, the WHFS applications use QUESTIONABLE data, but do not use BAD data. The QUESTIONABLE data should be reviewed to determine whether it is truly bad, which it often is, especially in the case of data deemed QUESTIONABLE because it failed the rate-of-change test. If the data are bad, the user should consider deleting the data using the Time Series application.

Note that if the shef_post_baddata token is set to REJECT, then all BAD data identified by the SHEF decoder will be sent to the RejectedData table (a.k.a. trashcan). Therefore, the window will only have QUESTIONABLE data, unless some local applications are setting physical element table data to BAD.

2.8 Rejected Data Viewer

The WHFS HydroView application provides a user interface to list the data contained in the RejectedData table. Data is placed in this table by the SHEF decoder because it was BAD data and the "switch" directs bad data to be written there, or because the user deleted it from the physical element tables using the Time Series Data Viewer. Local applications may also write to this table.

The RejectedData viewer provides the user with the option to move the data back into the physical element tables. Re-posting data in this manner results in the quality code being left unchanged. If the user wishes to update a BAD quality code, the Time Series Data Viewer can be used to do this. The window also allows the user to delete the data in the table immediately, rather than waiting for the data to be time-purged.

If the shef post_baddata token is set to PE, then all BAD data is sent to the physical element tables, and not the RejectedData table. Therefore, the RejectedData viewer will show only data that are manually deleted via the Time Series Data Viewer, unless local applications are also writing to the RejectedData table.

2.9 Time Series Review

The WHFS includes a powerful data viewing tool referred to as the Time Series Data Viewer. This application allows the user to view time series data in either graphical or tabular form. Because of possible problems with cluttering the graphs, the graphical form is limited in how many data attributes it can show, and therefore does not show the quality code information.

The tabular form does not have such limitations and is designed to show all the attributes associated with a value, including the data quality code. For each value, the tabular list shows the quality code as a single-letter - either G, Q, or B (GOOD, QUESTIONABLE, or BAD). When selecting an item from the list, a textual phrase below the list describes the value of the quality code. An example of the tabular time series window is given in Appendix B.

Both the graphical and the tabular modes of the time series allow the user to insert, update, and delete data. When deleting data, the deleted data is written to the RejectedData table, as mentioned earlier. When inserting data, the new value is assigned the default quality code value, which implies good. When modifying data, the quality code is set to GOOD, and it is noted specifically that its GOOD status is because it passed manual inspection. In addition, the SHEF qualifier code is set to M, for manually modified.

The Time Series application can be invoked via the Questionable/Bad Data Viewer window within HydroView. It is also available directly from the HydroView application, and as a stand-alone application. The Time Series application is described in a separate document.

2.10 Automated Checks

A scheduled cron job is provided as part of the standard WHFS installation to perform automated checks on the data. This mode of checking is most relevant for checks which require multiple-values, whether they be values for multiple times for one location, or for multiple locations for one time. To do these checks as each value is posted would not be meaningful because the necessary data would often not be available. Also, it would be extremely burdensome because of the typically large throughput of data entering the IHFS database. A timed cron job approach is used to allow the data to be posted and build up to a level that provides enough values to properly perform the quality test.

The cron job that is executed is the script:

/awips/hydroapps/whfs/standard/bin/run_alarm_whfs

This script operates by actually running two other scripts. The first performs automated checking of the data, for both quality controls and alert/alarm monitoring. The second is strictly for alert/alarm purposes; it generates and distributes a report of the current alert/alarm conditions. Despite its name, the run_alarm_whfs script does perform quality control operations via the first script it executes:

/awips/hydroapps/whfs/standard/bin/run_roc_checker

This job performs rate-of-change checks for the observed physical element tables specified in the script, and is run at a specified frequency. The default behavior is to run the checks on the Height and Precipitation (type-source of PC data only) tables every 10 minutes, and considering the data for the previous 6 hours. The script invokes the roc_checker program for one table per execution. Only data in the observed physical element tables are supported. In addition to the command-line arguments supplying the database name and the physical element table to process, the rate-of-change checker program supports optional arguments. These include:

an endtime as a relative or absolute time; default is now
the number of hours to process; default is 6 hours
a specific location to process; default is all locations
a specific physical element to process; default is all physical elements within the given physical element table
log error messages only; default is to show all
use GOOD data, or GOOD and QUESTIONABLE data; default is to use GOOD data only

For details on the syntax of these arguments and for other helpful information, refer to the notes given within the run_roc_checker script.

A value which fails the rate-of-change check is considered to be a QUESTIONABLE value. However, quite often, the rate-of-change values identify data that is obviously bad. The user can review the data via the Questionable/Bad Data Viewer and delete the rate-of-change QUESTIONABLE data with the Time Series Data Viewer that is integrated with the Questionable/Bad Data Viewer.

If desired, more locally-developed automated checks can be incorporated into the script or a new cron job running a new automated check can be locally implemented. Again, the design is intended to provide local expansion of the delivered functionality.

2.11 Interactive Checks

On the diagram is a process denoted Interactive Checks. Although no true interactive checks are provided, the user can review and delete data via the Time Series Data Viewer. Local applications can include interactive checks within the design of the existing framework. Presumably, they would read data from the physical element tables and update the quality code as per some local user functions.

If desired, the "library" software for the Time Series Viewer application can be provided for incorporation within the local interactive application. This would be helpful when some interactive checks identifies suspicious data and the context of the particular data value could then viewed in full using the Time Series Viewer. The software design of the Time Series application allows local applications to easily invoke the full functionality it provides. Doing so would require minor programming effort by the local office. Local offices should contact the Hydrology Laboratory to discuss options in this regard.

2.12 Internally Generated Data

Data can be inserted into the IHFS database by means other than the SHEF decoder. If this is done, the quality code must be set properly to adhere to the operations discussed in Section 2. Specifically, the initial value of the quality code should be set, and then any subsequent quality control operations should modify the code value accordingly. If the quality code value is not set properly, then the applications which filter the data may not interpret it correctly. Sections 3 and 4 describes the structure and programmer interface for managing the quality code, respectively.

3. Quality Code Definition

The quality code is presented in most displays as one of three values; GOOD, QUESTIONABLE, or BAD. However, this is just the summary representation of a quality code which can contain a large amount of attributes describing the quality of a given value. This section provides a complete description of the fields within the quality code.

3.1 Overall Database Structure

In the IHFS database are a collection of tables that contain the physical data for locations. These tables segregate the data according to the SHEF data model. Each table corresponds to a type-source (given by the first letter of the SHEF type-source code) indicating whether the data are observed, forecast, contingency, or processed. The data are segregated further by its physical element (i.e. typically the first letter of the SHEF physical element code). The observed physical element tables in addition to the ProcValue table include: Agricultural, Discharge, Evaporation, Ground, Height, Ice, Lake, Moisture, GateDam, Pressure, Precip, CurPrecip, Radiation, Snow, Temperature, Weather, Wind, and Yunique. The forecast physical element tables in addition to the ContingencyValue table are: FcstDischarge, FcstHeight, FcstPrecip, FcstTemperature.

The physical element tables of the IHFS database follow a standard structure, where the observed and processed data have the same fields, and the forecast and contingency data have the same fields. As an example, the observed tables have the following structure:

lid	char(8)
pe	char(2)
dur	smallint
ts	char(2)
extremum	char(1)
obstime	datetime year to second
value	float
shef_qual_code	char(1)
quality_code	integer
revision	smallint
product_id	char(10)
producttime	datetime year to second
postingtime	datetime year to second
processed_code	smallint

There are two fields in the physical element tables that are directly related to the quality of the data. These are the shef_qual_code and the quality_code. The shef_qual_code is the SHEF qualifier code associated with each incoming data value. Its value is used to define the value of the quality code, as described in Section 2.

3.2 History of the Internal Quality Code

The IHFS quality_code in the database is the primary topic of this section. Although the quality code field existed in the database prior to Release 5.0, the structure and operations of this code have been completely redefined. Previously, this field had one of three values indicating whether the data value passed the gross range check (=2), failed the gross range check (=1), or the gross range check was not performed (=0).

3.3 Overview of Quality Code

In Release 5.0, the field can take on an almost infinite set of values. That is because the integer field is now treated in a bit-wise fashion, where each of the 32 bits are turned on and off to indicate specific characteristics of the data quality of the physical element value. This 32 bit word is divided into two parts: the summary bits, and the details bits.

The summary bits include the first 8 bits and are used to summarize results of various QC tests performed on the observations. The summary bits play an extremely important role in applications to quickly determine whether they wish to use a certain value, depending on its quality. It is from the summary bits that the applications can quickly determine if the value is GOOD, QUESTIONABLE, or BAD.

If applications wish to know precise details about the quality tests, the second part provides this information. The second part consists of the remaining 24 bits and is used to contain the results of the various quality control tests.

3.4 Quality Code Structure

The structure of the quality code is given below. Details of each field are given in the next section for the bits which are currently assigned.

Bit	Description
(Summary bits)	
31:	Sign bit, not used
30:	Value-Not-Bad Indicator
29:	Value-Not-Questionable Indicator
28:	Post Ingest Tests Performed Indicator
24-27:	Reserved for future use.

(Details bits)

23:	External QC Not-Bad Indicator
22:	Manual QC Good Indicator
21:	Gross Range Test Result
20:	External QC Not-Questionable Indicator
19:	Reasonableness Range Test Result
18:	Rate-of-change Test Result
8-17	Reserved for future use.
0 -7	Available for local use. (Please contact OHD for coordination purposes.)

Figure 2. Quality Code Bit Assignments

3.5 Summary Bits

The values of these bits should be consistent with the values of the details bits. For example, if the summary bits imply that it is a BAD or QUESTIONABLE value, there should be at least one details bit that indicates why it is considered as such.

<u>Bit</u> <u>Description</u>

31: Sign bit, not used, always set to 0.

30: Value-Not-BAD Indicator

1 = Indicates that the value is GOOD or QUESTIONABLE; i.e. it is not BAD. Only external codes or internal tests that have actually been checked or performed are reflected in this value; a value labeled as not BAD could still be bad. This is the default value.

0= Indicates the value is BAD. Data can be considered BAD for a number of reasons. External data quality indicators could state it was BAD “on arrival” (bit 23=0), it could be deemed bad by manual inspection (bit 22=0), or the value could have failed the gross range check (bit 21=0). The external SHEF qualifier codes which result in the value being BAD are the qualifiers B (bad) and R (rejected).

29: Value-Not-QUESTIONABLE Indicator

1 = Indicates that the value is not QUESTIONABLE. Only external codes or internal tests that have actually been checked or performed are reflected in this value; a value labeled as not QUESTIONABLE could still be questionable. This is the default value.

0 = Indicates the value is QUESTIONABLE. This is because it failed at least one check or test that questions its validity. Possible reasons include a questionable incoming SHEF qualifier code of F or Q (bit 20=0), a failed reasonable range test (bit 19=0), or a failed rate-of-change test (bit 18=0).

Because of the critical nature of the above two bits in summarizing the overall quality of the value, the possible combinations of the two bits and their meaning are given below:

Not-Bad	Not-Questionable	Meaning
1	1	GOOD
1	0	QUESTIONABLE
0	1	BAD
0	0	Not used; illogical since this would mean that the value is BAD and QUESTIONABLE!

Figure 3. Primary Summary Bit Combinations

28: Post Ingest Tests Run indicator.

1 = Indicates tests other than the SHEF ingest tests were run. This bit is currently not set or used.

0 = Indicates tests other than the SHEF ingest tests were not run. This is the default value.

3.6 Details Bits

For all the details bits, a binary value of “0" or “1" is used to indicate the status or outcome of each test or check. “1" indicates “passed” or “not run”, while 0 indicates “fail”. For all the details bits, the default value is set to 1.

Bit Description

23: External QC Not-BAD indicator

1 = Indicates that none of the external QC indicators associated with the value upon ingest into the IHFS indicate the value is BAD. Therefore, the external QC is either GOOD or QUESTIONABLE. The External QC Not-QUESTIONABLE bit (bit 20) indicates which is the case.

0 = Indicates that the value was identified as bad by some external QC field. If the value of this bit is 0, the summary bit Value-Not-BAD (bit 30) must also be set to 0. Because SHEF data is the normal manner in which data enters the IHFS database, the SHEF qualifier code generally controls the value of this bit. A SHEF qualifier code value of “B” (bad) or “R” (rejected) causes this bit to be set to 0.

22: Manual QC GOOD Indicator

1 = Indicates that the value was determined to be GOOD via manual review. This value may result from some manual intervention or may simply result from the default state, i.e. just because it is a value of 1 does not mean it has been manually reviewed.

0 = Indicates that the value has been determined to be either BAD or QUESTIONABLE via manual review. The value of the summary bits for Value-Not-Bad (bit 30) and Value-Not-Questionable (bit 29) indicate which is the case. Anytime a BAD or QUESTIONABLE value is manually updated, the value of this bit reverts to 1.

21: Gross Range Test Result

1 = Indicates this value passed the gross range test, or the test was not performed.

0 = Indicates this value failed the gross range test. If this is the value of this bit, the summary bit Value-Not-BAD (bit 30) must also be set to 0, because values that fail this test are considered to be bad.

20: External QC Not-QUESTIONABLE Indicator

1 = Indicates that the external QC indicators associated with the value upon ingest into the IHFS indicate the value is not QUESTIONABLE. This bit works together with External QC Not-BAD bit (bit 23) to define the external QC characteristics of the data.

0 = Indicates that the data was identified as questionable by some external QC field. If this is the value of this bit, the summary bit Value-Not-QUESTIONABLE (bit 29) must also be set to 0. Because SHEF data is the normal manner in which data enters the IHFS database, the SHEF qualifier code generally controls the value of this setting. A SHEF qualifier code value of “Q” (questionable) or “F” (flagged) causes this bit to be set to 0.

19: Reasonable Range Test Result

1 = Indicates this value passed the reasonable range test, or the test was not performed.

0 = Indicates this value failed the reasonable range test. If this is the value of this bit, the summary bit Value-Not-QUESTIONABLE (bit 29) must also be set to 0, because values that fail this test are considered to be questionable.

18: Rate-of-change Test Result

1 = Indicates this value passed the rate-of-change test, or the test was not performed.

0 = Indicates this value failed the rate-of-change test. If this is the value of this bit, the summary bit Value-Not-QUESTIONABLE (bit 29) must also be set to 0.

4. Programmers Reference

The Office of Hydrologic Development has developed a set of library functions to support the programming efforts related to the quality code operations. These library functions insulate the application programs from all the details and inter-bit relationships of the bit-wise quality code field.

In order to encourage local application development, these functions are being made available to field developers on the OHD web page. The software is written in the C programming language. The library provides two files: a ".c" file containing the functions, and a ".h" file containing the function prototypes and symbolic constants used in the functions, and which should be referenced by the calling functions.

This section serves as the programmer documentation for these functions. If you are not developing software which uses quality code operations, than this section can be skipped. If you are developing quality control related software, then the functions described in this section should be extremely helpful. In order to appreciate the nuances of these operations, the previous sections should be read. Because the library functions provide a high-level layer on top of the lower-level bit operations, one does not need be an expert on the bit settings described in Section 3. However, strong knowledge of the design concepts of the summary and details bit within the 32-bit field is helpful in understanding which functions, and which function arguments, to use.

4.1 Quality Code Read Implications

When reading data, it may be the case that the application wishes to reject all bad data. Or the application may want to consider questionable data. Or maybe an application only wants to read bad data for review purposes. Regardless, the two summary bits provided in the highest-significant bit positions (not counting the sign bit), allow these checks to be made easily by treating the quality code as the long integer it is, and only reading data with a quality value within a certain numeric range. The numeric range can be easily managed in the SQL query used to retrieve the data in the library functions.

If an application needs to determine the results of a specific test, it is generally not practical to specify a numeric range for the data retrieval. This is because the other bits within the quality code can "push" the data to almost any numeric range since the bit in question is only one of 31 bits that define the overall integer value. If looking for data with a specific bit setting this should be done using a customized check in the application program. The library functions can limit the retrieval based on the three levels of data, but any further search for particular bit conditions must be done by the customized check. Library functions can be used to facilitate this customized check.

To retrieve data for a specific level(s) of data quality, the library function build_qc_where() can be used to create the SQL SELECT ... WHERE sub-clause for defining the appropriate numeric range. This sub-clause can be appended to the rest of the application's SQL SELECT statement or ESQL FETCH statement.

Once the data are retrieved, the application can check if the quality code meets some condition by use of the library function check_qccode(). If the application wishes to check the value of a specific bit, then the check_qcbit() function can be used.

4.2 Quality Code Write Implications

Library functions are available to set the value of the quality code. Normally, when a data value is first "created", the quality control code is set to an initial value. Then it may be modified further. Typically this is done by setting one or more bits together. Because the default setting for the details bits is 1, or passed, one may choose not to set the value if the subsequent check shows that indeed it passed. However, if the value failed the check, then not only must the appropriate details bit be set to false, then the associated summary bit should be set also. The set_qccode() function allows the user to set the quality code in a way that may set two or all bits simultaneously.

If only one bit needs to be set, such as is the case when setting a test result to good, then the set_qcbit() function can be used. Note that just because the value passes one test, it does not mean that the summary bits can be set to indicate, say, a GOOD value. That is because there may be another test that failed, and all it takes is one test to fail for a value to be deemed QUESTIONABLE or BAD.

4.3 Quality Code Interpretations

The quality code is a bit-wise defined number and therefore is difficult to interpret in simple terms. Given the value of the quality code, the library function build_qc_descr() provides a brief narrative type description, while the library function build_qc_symbol() provides a single character summarizing the level of the quality code as either good, questionable, or bad.

4.4 Symbolic Constants

Symbolic constants are used throughout the library functions internally, and can and should be used by external applications which call the library functions.

First, there is a set of constants which identify the bit positions within the 32-bit quality code. These names must be used when calling the set_qcbit() and the check_qcbit() functions. The names and values of these are:

SIGN_QC =	31
CERTAINTY_QC =	30
NOTQUEST_QC =	29
TEST_RUN_QC =	28
BIT27...BIT24 =	corresponding number
EXTERN_QC =	23
MANUAL_QC =	22
GROSSRANGE_QC =	21
EXTERN_NOTQUEST_QC =	20
REASONRANGE_QC =	19
ROC_QC =	18
BIT17...BIT0 =	corresponding number

Certain functions return a string argument to the calling function. In order to ensure that the calling function has allocated enough space for the value of these strings, it is recommended that the calling function use the following symbolic constants when declaring the string variable. The library function makes use of this same symbolic constant and thereby the calling function is ensured that it will have enough space if it uses the same string length constants. These constants include the character required for the terminating null

MAXLEN_QCWHERE	= 40	[used by build_qcwhere()]
MAXLEN_QCDESCR	= 120	[used by build_qcdescr()]
MAXLEN_QCSYMBOL	= 2	[used by build_qcsymbol()]

These symbolic constants are used for requesting a check of a value in the check_qccode() function or when building a SQL WHERE sub-clause using the function build_qcwhere(). Their values are not important, as they are only used for unique identification purposes.

QC_PASSED
QC_QUESTIONABLE
QC_FAILED

QC_NOT_PASSED
QC_NOT_FAILED

QC_ROC_PASSED

The symbolic constant QC_DEFAULT is used for setting the quality code to an initial value.

These requests are for setting the qc code via the set_qccode() function. For most requests, this functions sets a specific bit and its associated summary bit. For each of the symbolic constants listed below, the changes to the bits are listed, using the symbolic constants for the bit identifiers.

QC_EXTERN_QUEST -	EXTERN_NOTQUEST_QC=0 NOTQUEST_QC=0 (summary)
QC_EXTERN_FAILED -	EXTERN_QC=0 CERTAINTY_QC=0 (summary)
QC_GROSSRANGE_FAILED -	GROSSRANGE_QC=0 CERTAINTY_QC=0 (summary)
QC_REASONRANGE_FAILED -	REASONRANGE_QC=0 NOTQUEST_QC=0 (summary)
QC_ROC_FAILED -	ROC_QC=0 NOTQUEST_QC=0 (summary)
QC_MANUAL_PASSED -	QC_DEFAULT (sets all bits)
QC_MANUAL_QUEST -	MANUAL_QC=0 CERTAINTY_QC=1 (summary) NOTQUEST_QC= 0 (summary)
QC_MANUAL_FAILED -	MANUAL_QC=0 CERTAINTY_QC=0 (summary)
QC_MANUAL_NEW -	QC_DEFAULT (sets all bits)

These symbolic constants represent return status codes.

FALSE_QC =	0
TRUE_QC =	1
INVALID_QC_REQUEST =	-1
VALID_QC_REQUEST =	1

4.5 Special Bit Patterns

For those wishing to perform simple SQL queries to determine which level the data quality is assigned, there are some bit settings that deserve special note. The bit pattern for certain quality code values and their numeric equivalent are listed below.

DEFAULT_QC_VALUE:

This value will be seen often.

Bit pattern 0110 1111 1111 1111 1111 1111 1111 1111 = 1,879,048,191.

GOOD_QUESTIONABLE_THRESHOLD:

A quality code value indicates GOOD if it is greater than or equal to this value.
Bit pattern 0110 0000 0000 0000 0000 0000 0000 0000 = 1,610,612,736.

QUESTIONABLE_BAD_THRESHOLD:

A quality code value indicates BAD if it is less than this value; therefore a value is QUESTIONABLE if it is >= QUESTIONABLE_BAD threshold and < GOOD_QUESTIONABLE threshold.
Bit pattern 0100 0000 0000 0000 0000 0000 0000 0000 = 1,073,741,824

ALL_ONES

This value is currently not used, but is interesting to point out.
Bit pattern 0111 1111 1111 1111 1111 1111 1111 1111 = 2,147,483,647 = 2**31 - 1

4.6 Function Prototypes and Descriptions

void build_qc_where(int request, char *qc_where_subclause);

Used for facilitating a data read based on the quality code. Given a request, the function returns a portion of the "where" portion of the SQL SELECT statement. For example, if the user is interested in defining a where sub-clause for a retrieval of data that is questionable, the call would read:

```
build_qc_where(QC_QUESTIONABLE, where);
```

and the value of the returned where clause would be "quality code between #1 and #2", where #1 and #2 are the upper and lower numeric limits of the quality code which includes the range for questionable data. This range is easily defined by considering the value of the two summary bits Value-Not-BAD and Value-Not-QUESTIONABLE. The returned where clauses can be appended onto a SQL SELECT statement in the calling function.

The supported values of the request are: QC_PASSED, QC_NOT_PASSED, QC_FAILED, QC_QUESTIONABLE, QC_NOT_FAILED. All other requests will result in a error message written for the WHERE sub-clause. The calling function should use a character string which is allocated to at least the size of the symbolic constant MAX_QCWHERE.

void build_qc_descr(long quality_code, char *qc_descr);

Used for interpreting the quality code value. This function returns a textual description of the quality code, given the value of the quality code. For example, if the quality_code indicates a questionable value because it failed the external QC check, the returned string would be “Questionable: Failed External Test”. The calling function should use a character string which is allocated to at least the size of the symbolic constant MAX_QCDESCR.

void build_qc_symbol(long quality_code, char *qc_symbol);

Used for interpreting the quality code value. This function returns a single-character indicating the quality as per the three quality levels, given the value of the quality code. The returned value is either” G for GOOD, Q for QUESTIONABLE, and B for BAD. The calling function should use a character string which is allocated to at least the size of the symbolic constant MAX_QCSYMBOL.

int check_qccode(int bit_grp_descr, long quality_code);

Used for checking if the given quality code value meets some given descriptive state. The supported values of the given request are:QC_DEFAULT, QC_PASSED, QC_QUESTIONABLE, QC_ROC_PASSED, QC_FAILED, QC_QC_NOT_FAILED, QC_NOT_PASSED. This function is helpful for screening values that have already been retrieved from the database; i.e. they are in memory; to determine if they should be used for the user operation.

The function return argument value is either of the symbolic constants TRUE_QC or FALSE_QC, which are defined as 1 and 0 to allow for easy logic testing.

int set_qccode(int bit_grp_descr, long *quality_code);

Used for setting the given quality code as a whole, as opposed to setting one single bit within the code. When setting the code, the functions always sets the detail bit and the corresponding summary bit(s). The supported values of the given request are: QC_DEFAULT, QC_GROSSRANGE_FAILED, QC_REASONRANGE_FAILED, QC_ROC_FAILED, QC_EXTERN_FAILED, QC_EXTERN_QUEST, QC_MANUAL_PASSED, QC_MANUAL_QUEST, QC_MANUAL_FAILED, QC_MANUAL_NEW.

The function return argument is either of the symbolic constants VALID_QC_REQUEST or INVALID_QC_REQUEST indicator. If INVALID_QC_REQUEST is returned, then the bit_grp_descr request is unsupported and should be corrected.

```
int check_qcbit(int     bit_position,  long    quality_code);
```

Used for checking if the given bit in the given quality code is set to True or False. The function return argument value is either of the symbolic constants TRUE_QC or FALSE_QC, which are defined as 1 and 0 to allow for easy logic testing.

```
int set_qcbit(int       bit_position,  int       setting,  long      *quality_code);
```

Used for setting the given bit of the given quality code to the given setting, where the setting is either TRUE_QC or FALSE_QC. If setting a details bit, always remember to set the corresponding summary bit. The function return argument is either of the symbolic constants VALID_QC_REQUEST or INVALID_QC_REQUEST indicator. If INVALID_QC_REQUEST is returned, then the bit position is unsupported and should be corrected.

Appendix A. Diagram of QC Operations in the IHFS

Appendix B. Example Displays from QC Related Applications

On the following pages are screen dumps of the following graphical displays:

1. QC/Alert/Alarm Limits Manager -

   This display is from the HydroBase application and shows the window used to manage the limits which are defined for the QC thresholds, in addition to the limits used for alert/alarm purposes.

2. Questionable/Bad Data Viewer -

   This display is from the HydroView application and shows the window which lists all the data which is either questionable or bad.

3. Rejected Data Viewer -

   This display is from the HydroView application and shows the window which lists all the data in the RejectedData table, which serves as the "trash can" for rejected or deleted data.

4. Time Series Tabular Data Viewer -

   This display is from the Time Series application, which can be run in a stand-alone mode, or can be invoked from various windows in multiple programs, including the Questionable/Bad Data Viewer window. This display is the general tabular list for data in the physical element tables.